Administración de servidores

Remo Suppi Boldrito

P07/M2103/02286



Índex

Introducción

La interconexión entre máquinas y las comunicaciones de alta velocidad han permitido que los recursos que se utilicen no estén en el mismo sitio geográfico del usuario. UNIX (y por supuesto GNU/Linux) es probablemente el máximo exponente de esta filosofía, ya que desde su inicio ha fomentado el compartir recursos y la independencia de 'dispositivos'. Esta filosofía se ha plasmado en algo común hoy en día, que son los servicios. Un servicio es un recurso (que puede ser universal o no) y que permite bajo ciertas condiciones obtener información, compartir datos o simplemente procesar la información remotamente. Nuestro objetivo es analizar los servicios que permiten el funcionamiento de una red. Generalmente, dentro de esta red existirá una máquina (o varias, según las configuraciones) que posibilitará el intercambio de información entre las restantes. Estas máquinas se denominan servidores y contienen un conjunto de programas que permiten que la información esté centralizada y sea fácilmente accesible. Estos servicios propician la reducción de costes y amplían la disponibilidad de la información, pero se debe tener en cuenta que un servicio centralizado presenta inconvenientes, ya que puede quedar fuera de línea y dejar sin servicio a todos los usuarios.

Una arquitectura de servidores debe tener los servicios replicados (*mirrors*) para solventar estas situaciones.

Los servicios se pueden clasificar en dos tipos: de vinculación ordenador-ordenador o de relación hombre-ordenador. En el primer caso se trata de servicios requeridos por otros ordenadores, mientras que en el segundo, son servicios requeridos por los usuarios (aunque hay servicios que pueden actuar en ambas categorías). Dentro del primer tipo se encuentran servicios de nombres, como el *domain name system* (DNS), el servicio de información de usuarios (NISYP), el directorio de información LDAP o los servicios de almacenamiento intermedio (*proxies*). Dentro de la segunda categoría, se incluyen servicios de conexión interactiva y ejecución remota (SSH, telnet), transferencia de ficheros (FTP), intercambio de información a nivel de usuario, como el correo electrónico (MTA, IMAP, POP), *news*, World Wide Web, *Wiki* y archivos (NFS). Para mostrar las posibilidades de GNU/Linux Debian-FC6, se describirán cada uno de estos servicios con una configuración mínima y operativa, pero sin descuidar aspectos de seguridad y estabilidad.

1. *Domain name system* (DNS)

La funcionalidad del servicio de DNS (como se explicó en la unidad dedicada a la administración de red) es convertir nombres de máquinas (legibles y fáciles de recordar por los usuarios) en direcciones IP o viceversa.

Ejemplo

A la consulta de cuál es el IP de pirulo.remix.com, el servidor responderá 192.168.0.1 (esta acción es conocida como *mapping*); del mismo modo, cuando se le proporcione la dirección IP, responderá con el nombre de la máquina (conocido como *reverse mapping*).

El *domain name system* (DNS) es una arquitectura arborescente para evitar duplicación de la información y facilitar la búsqueda. Por ello, un único DNS no tiene sentido sino como parte del árbol.

La aplicación que presta este servicio se llama *named*, se incluye en la mayoría de distribuciones de GNU/Linux (/usr/sbin/named) y forma parte de un paquete llamado *bind* (actualmente versión 9.x) coordinado por ISC (*Internet software consortium*). DNS es simplemente una base de datos, por lo cual es necesario que las personas que la modifiquen conozcan su estructura, ya que, de lo contrario, el servicio quedará afectado. Como precaución, debe tenerse especial cuidado en guardar las copias de los archivos para evitar cualquier interrupción en el servicio. El paquete sobre Debian se encuentra como bind y bind.doc. [LN01, Deb03c, IET03]. Las configuraciones son similares, son FC, pero necesitaréis instalar bind, bind-utils and caching-nameserver que serán gestionadas por el yum por ejemplo.

1.1. Servidor de nombres caché

En primer lugar se configurará un servidor de DNS para resolver consultas que actúe como caché para las consultas de nombres (*resolver, caching only server*). Es decir, la primera vez consultará al servidor adecuado porque se parte de una base de datos sin información, pero las veces siguientes responderá el servidor de nombres caché, con la correspondiente disminución del tiempo de respuesta. Para configurar el servidor de nombres caché, se necesita el archivo /etc/bind/named.conf (en Debian), que tiene el siguiente formato (se han respetado los comentarios originales dentro del archivo, indicados por //):

```
options {
directory “/var/cache/bind”;
        // query-source address * port 53;
        // forwarders {
        //            0.0.0.0;
        //
```

```
        };
        auth-nxdomain no; # conform to RFC1035
        };
// prime the server with knowledge of the root servers}
zone "." {
        type hint;
        file "/etc/bind/db.root"; };
        // be authoritative for the localhost forward and reverse zones, and for
        // broadcast zones as per RFC 1912
        }
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
        };
zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
        };
zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
        };
zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
        };
// add entries for other zones below here
}
```

La sentencia directory indica dónde se encontrarán los archivos de configuración restantes (/var/cache/bind en nuestro caso). El archivo /etc/bind/db.root contendrá algo similar a lo siguiente (se muestra sólo las primeras líneas que no son comentarios indicados por ';' y se debe tener cuidado con los puntos (.) al inicio de algunas líneas –se puede obtener directamente de Internet actualizado–):

```
...
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
;
; formerly C.PSI.NET
;
. 3600000 NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 A 192.33.4.12
;
...
```

Este archivo describe los *root name servers* en el mundo. Estos servidores cambian, por lo que se debe actualizar periódicamente el archivo. Las siguientes secciones son las *zone*; las *zones localhost* y 127.in-addr.arpa, que se vinculan a los ficheros al archivo etc/bind/db.local y /etc/bind/db.127, se refieren a la resolución directa e inversa para la interfaz local. Las *zones* siguientes son para las zonas de difusión (según RFC 1912) y al final se deberían agregar las propias. Por ejemplo, el archivo db.local podría ser (';' significa 'comentario'):

```
; BIND reverse data file for local loopback interface
$TTL 604800
@       IN      SOA     ns.remix.bogus.         root.remix.bogus. (
        1               ; Serial
        604800 ;         Refresh
        86400           ; Retry
        2419200;         Expire
        604800);        Negative Cache TTL
@       IN      NS      ns.remix.bogus.
1.0.0   IN      PTR     localhost.
```

Explicaremos su utilización más adelante. Lo siguiente es poner como *name server* en el /etc/resolv.conf:

```
search subdominio.su-dominio.dominio su-dominio.dominio
# por ejemplo search remix.bogus bogus
nameserver 127.0.0.1
```

Donde se deberán reemplazar los subdominio.su-dominio.dominio por los valores adecuados. La línea *search* indica qué dominios se buscarán para cualquier *host* que se quiera conectar (es posible sustituir *search* por *domain*, aunque tienen comportamientos diferentes) y el *nameserver* especifica la dirección de su *nameserver* (en este caso su propia máquina, que es donde se ejecutará el *named*). El *search* tiene el siguiente comportamiento: si un cliente busca la máquina pirulo, primero se buscará pirulo.subdominio.su-dominio.dominio, luego pirulo.su-dominio.dominio y finalmente, pirulo. Esto implica tiempo de búsqueda; ahora bien, si se tiene la seguridad de que pirulo está en subdominio.su-dominio.dominio, no es necesario poner los restantes.

El paso siguiente es poner en marcha el *named* y mirar los resultados de la ejecución. Para poner en marcha el *daemon*, se puede hacer directamente con el *script* de inicialización `/etc/init.d/bind9 start` (en caso de que el *named* ya se esté ejecutando, hacer `/etc/init.d/bind9 reload`) o, si no, `/usr/sbin/named`. Mirando el *log* del sistema en /var/log/daemon.log veremos algo como:

```
Sep 1 20:42:28 remolix named[165]: starting BIND 9.2.1 \\
Sep 1 20:42:28 remolix named[165]: using 1 CPU \\
Sep 1 20:42:28 remolix named[167]: loading configuration from '/etc/bind/named.conf'
```

Aquí se indica el arranque del servidor y los mensajes de errores (si los hay), los cuales se deberán corregir y volver a empezar. Ahora se puede verificar la configuración con comandos como nslookup (original, fácil pero obsoleto según algunos autores), host o dig (recomendado). La salida de `dig -x 127.0.0.1` será algo como:

```
# dig -x 127.0.0.1
;; <<>> DiG 9.2.1 <<>> -x 127.0.0.1
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31245
```

```
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION: ;1.0.0.127.in-addr.arpa. IN PTR
;;ANSWER SECTION: 1.0.0.127.in-addr.arpa. 604800 IN PTR localhost.
;; AUTHORITY SECTION: 127.in-addr.arpa. 604800 IN NS ns.remix.bogus.
;; Query time: 1 msec
;; SERVER: 127.0.0.1 #53(127.0.0.1)
;; WHEN: Mon Sep 1 22:23:35 2003
;; MSG SIZE rcvd: 91
```

Donde se puede ver que la consulta ha tardado 1 milisegundo. Si se dispone de conexión a Internet, se podría buscar alguna máquina dentro de vuestro dominio y ver el comportamiento de vuestro servidor. En BIND9 existe el lwresd (*lightweight resolver daemon*), que es el *daemon* que provee servicios de nombres a clientes que utilizan la biblioteca de BIND9 *lightweight resolver*. Es esencialmente un servidor caché (como el que se ha configurado) el que realiza las consultas utilizando el BIND9 *lightweight resolver protocol* en lugar del protocolo DNS. Este servidor escucha por la interfaz 127.0.0.1 (por lo cual, sólo atiende a procesos de la máquina local) en UDP y el puerto 921. Las consultas de los clientes son decodificadas y resueltas utilizando el protocolo DNS. Cuando se obtienen las respuestas, el lwresd las codifica en el formato lightweight y las retorna al cliente que las ha solicitado.

Por último, como ya se ha mencionado, el *kernel* utiliza diversas fuentes de información, que para la red se obtienen desde /etc/nsswitch.conf. Este archivo indica desde dónde obtener la fuente de información y para el caso de los nombres de máquinas e IP hay una sección como:

hosts: files dns

Esta línea (si no existe se debe agregar) indica que quien necesite un nombre de una máquina o una IP consulte primero en /etc/hosts y luego en DNS de acuerdo a los dominios indicados en /etc/resolv.conf.

1.2. *Forwarders*

En redes con una considerable carga, es posible equilibrar el tráfico utilizando la sección de *forwarders*. Si vuestro proveedor de red (ISP) tiene uno o más *nameservers* estables, es recomendable utilizarlos para descongestionar las consultas sobre su servidor. Para ello, debe quitarse el comentario (//) de cada línea de la sección *forwarders* del archivo /etc/bind/named.conf y reemplazar el 0.0.0.0 con las IP de los *nameservers* de su ISP. Esta configuración es recomendable cuando la conexión es lenta, por ejemplo, por módem.

1.3. Configuración de un dominio propio

DNS posee una estructura en árbol y el origen es conocido como '.' (ver /etc/bind/db.root). Bajo el '.' existen los TLD (*top level domains*) como org, com, edu, net, etc. Cuando se busca en un servidor, si éste no conoce la res-

puesta, se buscará recursivamente en el árbol hasta encontrarla. Cada '.' en una dirección (por ejemplo, pirulo.remix.com) indica una rama del árbol de DNS diferente y un ámbito de consulta (o de responsabilidad) diferente que se irá recorriendo en forma recursiva de izquierda a derecha.

Otro aspecto importante, además del dominio, es el in-addr.arpa (*inverse mapping*), el cual también está anidado como los dominios y sirve para obtener nombres cuando se consulta por la dirección IP. En este caso, las direcciones son escritas al revés, en concordancia con el dominio. Si pirulo.remix.com es la 192.168.0.1, será escrita como 1.0.168.192, en concordancia con pirulo.remix.com.

A continuación, configuraremos el dominio propio remix.bogus en el archivo /etc/bind/db.127 [LN01]:

```
; BIND reverse data file for local loopback interface
$TTL 604800
@ IN SOA ns.remix.bogus. root.remix.bogus. (
        1                   ; Serial
        604800              ; Refresh
        86400               ; Retry
        2419200             ; Expire
        604800 )            ; Negative Cache TTL
@       IN      NS          ns.remix.bogus.
1.0.0   IN      PTR         localhost.
```

Se debe tener en cuenta el '.' al final de los nombres de dominio. El origen de la jerarquía de una *zone* está especificada por la identificación de la zona, en nuestro caso 127.in-addr.arpa. Este archivo (db.127) contiene 3 registros: SOA, NS, PTR. El SOA (*start of authority*) debe estar en todos los archivos de zona al inicio, después de TTL, y el símbolo @ significa el origen del dominio; NS, el servidor de nombres para el dominio, y PTR (*domain name pointer*), que es el *host* 1 en la subred (127.0.0.1) y se denomina *local host*. Éste es el archivo serie 1 y el responsable del mismo es root@remix.bogus (último campo de la línea SOA). Ahora se podría reiniciar el *named* de la forma antes indicada y con el `dig -x 127.0.0.1`, ver su funcionamiento (que sería idéntico al mostrado anteriormente).

A continuación, habrá que añadir una nueva zona en el named.conf:

```
zone "remix.bogus" {
        type master;
        notify no;
        file "/etc/bind/remix.bogus";
        };
```

Se debe recordar que en el named.conf los dominios van sin el '.' final. En el archivo remix.bogus se pondrán los *hosts* de los cuales seremos responsables:

```
; Zone file for remix.bogus
$TTL    604800
@       IN      SOA         ns.remix.bogus. root.remix.bogus. (
        199802151           ; serial, todays date + todays serial
        604800              ; Refresh
        86400               ; Retry
```

```
        2419200             ; Expire
        604800 )            ; Negative Cache TTL
@       NS      ns          ; Inet Address of name server
        MX      10          mail.remix.bogus. ; Primary Mail Exchanger
localhost       A           127.0.0.1
ns              A           192.168.1.2
mail            A           192.168.1.4
                TXT "Mail Server"
ftp             A           192.168.1.5
                MX          10 mail
www             CNAME       ftp
```

Aquí aparece un nuevo registro MX que es el Mail eXchanger. Es el lugar donde se enviarán los correos electrónicos que lleguen, alguien@remix.bogus, y será a mail.remix.bogus (el número indica la prioridad si tenemos más de un MX). Recordar el '.' necesario siempre en los archivos de *zone* al final del dominio (si no se ponen, el sistema agrega el dominio SOA al final, lo cual transformaría, por ejemplo, mail.remix.bogus en mail.remix.bogus.remix.bogus, que es incorrecto). CNAME (*canonical name*) es la forma de dar a una máquina uno o varios alias. A partir de ahora se estaría en condiciones (después de /etc/init.d/bind9 reload) de probar, por ejemplo, `dig www.remix.bogus`.

El último paso es configurar la zona inversa, es decir, para que pueda convertir direcciones IP en nombres, por ejemplo, agregando una nueva zona:

```
zone "192.168.1.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/192.168.1";
        };
```

Y el archivo /etc/bind/192.168.1 similar al anterior:

```
$TTL 604800
@       IN      SOA         ns.remix.bogus. root.remix.bogus. (
        199802151           ; serial, todays date + todays serial
        604800              ; Refresh
        86400               ; Retry
        2419200             ; Expire
        604800 ) ; Negative Cache TTL
@       NS      ns.remix.bogus.
2       PTR     ns.remix.bogus
4       PTR     mail.remix.bogus
5       PTR     ftp.remix.bogus
```

Éste se podría probar nuevamente con `dig -x 192.168.1.4`. Se debe tener en cuenta que estos ejemplos están sobre IP privadas, es decir, no IP de Internet. Otra cuestión importante es no olvidar el notify no, ya que si no nuestros experimentos con DNS se propagarán a los servidores del árbol de DNS (incluso modificando los DNS de nuestro proveedor u institución). Sólo se debe modificar cuando estamos seguros de que funciona y queremos propagar los cambios. Para ver un ejemplo real, consultar DNS-HOWTO en http://tldp.org/HOWTO/DNS-HOWTO-7.html.

Una vez creado un máster *server*, debe crearse un *slave server* por seguridad, que es idéntico al máster, excepto que la zona en lugar de *type master* deberá tener *slave* y la IP del *master*. Por ejemplo:

```
zone "remix.bogus" {
        type slave;
        notify no;
        masters {192.168.1.2; }
        };
```

2. NIS (YP)

Con el fin de facilitar la administración y dar comodidad al usuario, en redes de diferentes tamaños que ejecutan GNU/Linux (o Sun o algún otro sistema operativo con soporte para este servicio) se ejecutan servicios de Network Information Service, NIS (o *yellow pages,* YP, en la definición original de Sun). GNU/Linux puede dar apoyo como cliente/servidor de NIS y pueden actuar como cliente (versión “beta”) de NIS+, que es una versión más segura y optimizada de NIS. La información que se puede distribuir en NIS es: usuarios (*login names*), palabras de acceso (*passwords*) (/etc/passwd), directorios de usuario (*home directories*), información de grupos (*group information*) (/etc/group), lo cual presenta la ventaja de que, desde cualquier máquina cliente o desde el mismo servidor, el usuario se podrá conectar con la misma cuenta y *password* y al mismo directorio (aunque el directorio deberá ser montado anteriormente sobre todas las máquinas clientes por NFS o utilizando el servicio de *automount*). [Miq01, Kuk03]

La arquitectura NIS es del tipo cliente-servidor, es decir, existe un servidor que dispondrá de todas las bases de datos y unos clientes que consultarán estos datos en forma transparente para el usuario. Por ello, se debe considerar la configuración de servidores ‘de refuerzo’ (llamados secundarios) para que los usuarios no queden bloqueados ante la caída del servidor principal. Es por ello por lo que la arquitectura realmente se denomina de múltiples servidores (*master+mirrors*-clientes).

2.1. ¿Cómo iniciar un cliente local de NIS en Debian?

Un cliente local significa anexar el ordenador a un dominio NIS ya existente:

- Primero se debe verificar que se tienen instalados los paquetes *netbase* (red básica TCP/IP), *portmap* (servidor que convierte números RPC en puertos DARPA y es necesario para programas que ejecutan RPC, incluyendo NFS y NIS), y *nis* (específico). Se recomienda usar el comando kpackage o directamente con apt-get (se puede verificar si está instalado con `apt-cache pkgnames`) en modo texto. El procedimiento de instalación del paquete NIS solicitará un dominio (NIS *domainname*). Éste es un nombre que describirá el conjunto de máquinas que utilizarán el NIS (no es un nombre de *host*). Tener en cuenta que NISPirulo es diferente que Nispirulo como nombre de dominio. Para configurar este dominio,

se puede utilizar el comando `nisdomainname`, dominio que se almacenará en /proc/sys/kernel/domainname.

- En primer lugar se debe iniciar el servicio portmap con:
  - `/etc/init.d/portmap start`
  - Se puede comprobar si estos servicios están activos con `rpcinfo -p`.
- Si el servidor NIS no es local, se deberá utilizar el comando ypbind. El comando ypbind es utilizado para encontrar un servidor para el dominio especificado, ya sea mediante *broadcast* (no aconsejado por inseguro) o buscando al servidor indicado en el archivo de configuración /etc/yp.conf (recomendable). El archivo /etc/yp.conf tiene la siguiente sintaxis:
  - *domain nisdomain server hostname*: indica utilizar el *hostname* para el dominio *nisdomain*. Se podrían tener más de una entrada de este tipo para un único dominio.
  - *domain nisdomain broadcast*: indica utilizar *broadcast* sobre la red local para descubrir un servidor de dominio *nisdomain*.
  - *ypserver hostname*: indica utilizar *hostname* como servidor. Es recomendable utilizar esta línea (*ypserver*) donde se deberá introducir la dirección IP del servidor NIS. Si se indica el nombre asegúrese que se puede encontrar por DNS la IP o que la misma figura en el archivo /etc/hosts ya que, de otro modo, el cliente se bloqueará.
- Inicie el servicio ejecutando:
  - `/etc/init.d/nis stop`

  y luego:
  - `/etc/init.d/nis start`
- A partir de este momento, el cliente NIS estará funcionando (se puede confirmar con `rpcinfo -u localhost ypbind`, que mostrará las dos versiones del protocolo activo) o se puede utilizar el comando `ypcat mapname` (por ejemplo, ypcat passwd, que mostrará los usuarios NIS definidos en el servidor) donde la relación *mapnames* a tablas de la base de datos NIS están definidos en /var/yp/nicknames.

2.2. ¿Qué recursos se deben especificar para utilizar en NIS?

Consideraremos que tenemos instalada una de las últimas distribuciones de Debian (por ejemplo, 3.0 Woody o 3.1 *Sarge*), que soporta la Libc6 (igualmente para FC4 o superior), y se quiere que los usuarios en una máquina cliente pue-

dan acceder a la información del servidor. En este caso, se debe orientar la consulta del *login* a las bases de datos adecuadas haciendo:

1) Verificar el fichero /etc/nsswitch.conf y asegurarse de que las entradas passwd, group, shadow y netgroup son similares a:

passwd: compat
group: compat
shadow: compat ...
netgroup: nis
Ver *man nsswitch.conf* para la sintaxis de este archivo.

2) Agregar la siguiente línea en las máquinas clientes NIS en el fichero /etc/passwd al final del archivo (indicará que si el usuario no es local, se lo preguntará al servidor de NIS):

+:::::: (un '+' y seis ':')

3) Debe tenerse en cuenta que en el /etc/passwd se puede utilizar el + y el **?** delante de cada nombre de usuario en el /etc/passwd para incluir/excluir el *login* de estos usuarios (*override*). Si se está utilizando *passwords* con *shadow* (más seguro, ya que no permite que un usuario normal pueda ver el *password* encriptado de otros usuarios) deberá incluir la siguiente línea al final del archivo /etc/shadow:

+:::::::: (un '+' y ocho ':')

4) Debe añadir también la siguiente línea al final de /etc/group:

+::: (un '+' y tres ':')

5) Las búsquedas de *hosts* (*hosts lookups*) se realizarán mediante DNS (y no por NIS), por lo cual, para aplicaciones Libc6 en el fichero /etc/nsswitch.conf habrá que cambiar la entrada hosts por la siguiente línea: `hosts: files dns`. O, si se prefiere hacer por NIS, `hosts: files nis`. Para aplicaciones Libc5, se deberá modificar el fichero /host.conf poniendo order hosts, dns o order hosts, nis según desee.

Con esta configuración se podrá realizar una conexión local (sobre el cliente NIS) a un usuario que no esté definido en el fichero /etc/passwd, es decir, un usuario definido en otra máquina (ypserver).

Por ejemplo, se podría hacer `ssh -l user localhost`, donde *user* es un usuario definido en ypserver.

2.3. ¿Cómo se debe ejecutar un *master* NIS *server*?

Consideramos que sobre la máquina se tiene instalado el paquete nis y el portmap (este último en funcionamiento) y que las bases de datos del NIS están creadas (ver el siguiente apartado):

- Habrá que asegurarse de que en el /etc/hosts se encuentran todas las máquinas que formarán parte del dominio en el formato FQDN (*fully qualified domain name*), que es donde se indican el *IP*, el nombre y dominio y el nombre sin dominio de cada máquina (por ejemplo, 192.168.0.1 pirulo.remix.com pirulo). Esto es necesario sólo en el *server*, ya que el NIS no utiliza DNS.

- Además, existe en el archivo /etc/defaultdomain con el nombre del dominio escogido. No utilicéis vuestro dominio DNS para no incurrir en un riego de seguridad, excepto que configuréis adecuadamente los archivos /etc/ypserv.securenets (que indica con un par netmask/network desde qué sitio se podrán conectar los clientes) y /etc/ypserv.conf (que realiza un control más detallado porque indica qué *hosts* pueden acceder a qué mapas, por ejemplo: passwd.byname o shadow.byname).

- Verificar que existe *NISSERVER* = *master* en /etc/default/nis.

- Por motivos de seguridad, se puede agregar el número de red local al archivo /etc/ypserv.securenets.

- Iniciar el servidor ejecutando el comando `/etc/init.d/nis stop` y luego `/etc/init.d/nis start`. Esta sentencia iniciará el *server* (ypserv) y el *password daemon* (yppasswdd), los cuales se podrán consultar si está activo con `ypwich -d domain`.

2.4. ¿Cómo se debe configurar un *server*?

La configuración del *server* se realiza mediante el comando `/usr/lib/yp/ypinit -m`; sin embargo, es necesario verificar que existe el archivo `/etc/networks`, que es imprescindible para este *script*.

Si este archivo no existe, cread uno vacío con `touch /etc/networks`. También se puede hacer que sobre el servidor se ejecute el cliente ypbind; así, todos los usuarios entran por NIS, como se indicó anteriormente, modificando el fichero /etc/passwd donde todas las entradas normales antes de la línea +:::::: serán ignoradas por el NIS (sólo podrán acceder localmente), mientras que las posteriores podrán acceder por el NIS desde cualquier cliente. [Miq01]

Considerad que a partir de este momento los comandos para cambiar el passwd o información de los usuarios como passwd, chfn, adduser no son válidos. En su lugar, se deberán utilizar comandos tales como yppasswd, ypchsh e ypchfn. Si se cambian los usuarios o se modifican los archivos mencionados, habrá que reconstruir las tablas de NIS ejecutando el comando `make` en el directorio /var/yp para actualizar las tablas.

Tener en cuenta que Libc5 no soporta shadow passwd, por lo cual no se debe utilizar *shadow* con NIS si se tienen aplicaciones con Libc5. No habrá ningún problema si se tiene Libc6 que acepta NIS con soporte *shadow*.

La configuración de un *slave server* es similar a la del *master* excepto que *NISSERVER = slave* en /etc/default/nis. Sobre el *master* se debe indicar que distribuya las tablas automáticamente a los *slaves* poniendo *NOPUSH = "false"* en el archivo /var/yp/Makefile.

Ahora se debe indicar al *master* quién es su esclavo ejecutando:

```
/usr/lib/yp/ypinit -m
```

y entrando los nombres de los *slave servers*. Esto reconstruirá los mapas, pero no enviará los archivos a los *slaves*. Para ello, sobre el *slave*, ejecutar:

```
/etc/init.d/nis stop

/etc/init.d/nis start
```

y, por último:

```
/usr/lib/yp/ypinit -s nombre_master_server.
```

Así el *slave* cargará las tablas desde el *master*.

También se podría poner en el directorio /etc/cron.d el archivo nis con un contenido similar a (recordar hacer un chmod 755 /etc/cron.d/nis):

```
20 * * * * root /usr/lib/yp/ypxfr_1perhour >/dev/null 2>&1
40 6 * * * root /usr/lib/yp/ypxfr_1perday >/dev/null 2>&1
55 6,18 * * * root /usr/lib/yp/ypxfr_2perday >/dev/null 2>&1
```

Con lo cual, se asegurará de que todos los cambios del *master* serán transferidos al servidor NIS *slave*.

Recomendación: Después de usar `adduser` para agregar un nuevo usuario sobre el servidor ejecutar `make -C /var/yp` para actualizar las tablas NIS (y siempre que se cambie alguna característica del usuario, por ejemplo la palabra clave con el comando passwd, que sólo cambiará el *password* local y no el de NIS). Para probar que el sistema está funcionando y el usuario dado de alta está en el NIS, puede hacer `ypmatch userid passwd` donde userid es el usuario dado de alta con adduser previamente y después de haber hecho el make.

Para verificar el funcionamiento del sistema NIS, se puede utilizar el *script* de http://tldp.org/HOWTO/NIS-HOWTO/verification.html, que permite una verificación más detallada sobre el NIS.

3. Servicios de conexión remota: telnet y ssh

3.1. Telnet y telnetd

Telnet es un comando (cliente) utilizado para comunicarse interactivamente con otro *host* que ejecuta el *daemon* telnetd. El comando telnet se puede ejecutar como telnet host o interactivamente como telnet, el cual pondrá el prompt "telnet>" y luego, por ejemplo: *open host*. Una vez establecida la comunicación, se deberá introducir el usuario y el *password* bajo el cual se desea conectar al sistema remoto. Se dispone de diversos comandos (en modo interactivo) tal como open, logout, mode (definir las características de visualización), close, encrypt, quit, set, unset, o puede ejecutar comando externos con '!'. Se puede utilizar el archivo /etc/telnetrc para definiciones por defecto, o .telnetrc para definiciones de un usuario particular (deberá estar en el directorio home de usuario).

El *daemon* telnetd es el servidor de protocolo telnet para la conexión interactiva. Telned es puesto en marcha generalmente por el *daemon* inetd y se recomienda incluir un wrapper tcpd (que utiliza las reglas de acceso en host.allow y el host.deny) en la llamada al telnetd dentro del archivo /etc/inetd.conf (por ejemplo, incluir una línea como:

telnet stream tcp nowait telnetd.telenetd /usr/sbin/tcpd /usr/bin/in.telnetd)

Para incrementar la seguridad del sistema, véase la unidad dedicada a la seguridad. En algunas distribuciones (Debian 3.0 o superiores), la funcionalidad de inetd se puede reemplazar por xinetd, que requiere que se configure el archivo /etc/xinetd.conf (ver la unidad dedicada a la de administración de red). Si igualmente se quiere poner en marcha inetd a modo de pruebas se puede utilizar la sentencia `/etc/init.d/inetd.real start`. Si el archivo /etc/uissue.net está presente, el telnetd mostrará su contenido al inicio de la sesión. También se puede utilizar /etc/security/access.conf para habilitar/deshabilitar *logins* de usuario, *host* o grupos de usuarios, según se conecten.

Se debe recordar que, si bien el par telnet-telnetd pueden funcionar en modo encrypt en las últimas versiones (transferencia de datos encriptados, aunque deben estar compilados con la opción correspondiente), es un comando que ha quedado en el olvido por su falta de seguridad aunque puede ser utilizado en redes seguras o situaciones controladas.

Si no está instalado se puede utilizar (Debian) apt-get install telnetd y después verificar que se ha dado de alta o bien en /etc/inetd.conf o en /etc/xinetd.conf (o en el directorio que estén definido los archivos por ejemplo /etc/xinetd.d según se indique en el archivo anterior con la sentencia include /etc/xinetd.d). O bien en el xinetd.conf o en el archivo /etc/xinetd.d/telentd deberá incluir una sección como (cualquier modificación en xinetd.conf deberá rearrancar el servicio con service xinetd restart):

```
service telnet
{
disable = no
flags = REUSE
socket_type = stream
wait = no
user = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
}
```

Se recomienda, en lugar de utilizar telnetd o bien utilizar SSL telnet(d) el cual reemplaza al telnet(d) utilizando encriptación y autentificación por SSL o bien utilizar SSH (siguiente sección). El SSLTelnet(d) puede funcionar con el telnet(d normal en ambas direcciones, ya que al inicio de la comunicación verifica si del otro lado (peer) soporta SSL y si no continúa con el protocolo telnet normal. Las ventajas con respecto al telnet(d) son que sus *passwords* y datos no circularán por la red en modo texto plano y nadie utilizando por ejemplo tcpdump podrá ver el contenido de la comunicación. También con SSLtelnet se puede utilizar para conectarse por ejemplo a un servidor web seguro (por ejemplo https://servidor.web.org) simplemente haciendo: `telnet servidor .web.org 443`.

3.2. SSH, *Secure shell*

Un cambio aconsejable hoy en día es utilizar ssh en lugar de telnet, rlogin o rsh. Estos comandos son inseguros (excepto SSLTelnet) por varias razones: la más importante es que todo lo que se transmite por la red, incluido *usernames* y *passwords*, es en texto plano (aunque existen versiones de telnet-telnetd encriptados, deben coincidir en que ambos lo sean), cualquiera que tenga acceso a esa red o a algún segmento de la misma puede obtener toda esta información y luego suplantar la identidad del usuario. La segunda es que estos puertos (telnet, rsh,...) es al primer lugar donde un *cracker* intentará conectarse. El protocolo ssh (en su versión OpenSSH) provee de un conexión encriptada y comprimida mucho más segura que, por ejemplo, telnet (es recomendable utilizar la versión 2 del protocolo). Todas las distribuciones actuales incorporan el cliente ssh y el servidor sshd por defecto.

3.2.1. ssh

Para ejecutar el comando, hacer:

ssh -l *login name host* o **ssh** *user@hostname*

A través de SSH se pueden encapsular otras conexiones como X11 o cualquier otra TCP/IP. Si se omite el parámetro -l, el usuario se conectará con el mismo usuario local y en ambos casos el servidor solicitará el passwd para validar la identidad del usuario. SSH soporta diferentes modos de autentificación (ver man ssh) basados en algoritmo RSA y clave pública.

Utilizando el comando `ssh-keygen -t rsa|dsa`, se pueden crear las claves de identificación de usuario. El comando crea en el directorio del .shh del usuario el fichero (por ejemplo, para el algoritmo de encriptación RSA) id_rsa y id_rsa.pub las claves privada y pública respectivamente. El usuario podría copiar la pública (id_rsa.pub) en la máquina remota en el directorio .ssh del usuario remoto, en el archivo authorized_keys. Este archivo podrá contener tantas claves públicas como sitios desde donde se quiera conectar a esta máquina en forma remota. La sintaxis es de una clave por línea y su funcionamiento es equivalente al archivo .rhosts (aunque las líneas tendrán un tamaño considerable). Después de haber introducido las claves públicas del usuario-máquina en este archivo, este usuario y desde esa máquina se podrá conectar sin *password*.

En forma normal (si no se han creado las claves), se le preguntará al usuario un *password*, pero como la comunicación será siempre encriptada, nunca será accesible a otros usuarios que puedan escuchar sobre la red. Para mayor información, consultar `man ssh`. Para ejecutar remotamente un comando, simplemente hacer:

ssh -l *login name host_comando_remoto*

Por ejemplo:

ssh -l user localhost ls -al

3.2.2. sshd

El sshd es el servidor (*daemon*) para el ssh (se puede instalar si no lo están con `apt-get install ssh` lo cual instala el servidor y el cliente). Juntos reemplazan al rlogin, telnet, rsh y proveen una comunicación segura y encriptada en dos *hosts* inseguros en la red.

Éste se arranca generalmente a través de los archivos de inicialización (/etc/init.d o /etc/rc) y espera conexiones de los clientes. El sshd de la mayoría de las dis-

tribuciones actuales soporta las versiones 1 y 2 del protocolo SSH. Cuando se instala el paquete, crea una clave RSA específica del *host*, y cuando el *daemon* se inicia, crea otra, la RSA para la sesión, que no se almacena en el disco y la cambia cada hora. Cuando un cliente inicia la comunicación, el cliente genera un número aleatorio de 256 bits que es encriptado con las dos claves del servidor y enviado. Este número se utilizará durante la comunicación como clave de sesión para encriptar la comunicación que se realizará a través de un algoritmo de encriptación estándar. El usuario puede seleccionar cualquiera de los disponibles ofrecidos por el servidor. Existen algunas diferencias (más seguro) cuando se utiliza la versión 2 del protocolo. A partir de este momento, se inician algunos de los métodos de autentificación de usuario descritos en el cliente o se le solicita el *password*, pero siempre con la comunicación encriptada. Para mayor información, consultar `man sshd`.

3.2.3. Túnel sobre SSH

Muchas veces tenemos un acceso a un servidor sshd, pero por cuestiones de seguridad no a otros servicios que no son encriptados (por ejemplo un servicio de consulta de mail POP3 o un servidor de ventanas X11) o simplemente se quiere conectar a un servicio a los cuales sólo se tiene acceso desde el entorno de la empresa. Para ello es posible establecer un túnel encriptado entre la máquina cliente (por ejemplo con Windows, y un cliente ssh llamado putty de software libre) y el servidor con sshd. En este caso, al vincular el túnel con el servicio, el servicio verá la petición como si viniera de la misma máquina. Por ejemplo, si queremos establecer una conexión para POP3 sobre el puerto 110 de la máquina remota (y que también tiene un servidor sshd) hacemos:

```
ssh -C -L 1100:localhost:110 usuario-id@host
```

Este comando pedirá el *password* para el usuario-id sobre *host*, y una vez conectado se habrá creado el túnel. Cada paquete que se envíe a la máquina local sobre el puerto 1100 será enviado a la máquina remota *localhost* sobre el puerto 110, que es donde escucha el servicio POP3 (la opción -C comprime el tráfico por el túnel).

Hacer túneles sobre otros puertos es muy fácil. Por ejemplo, supongamos que sólo tengamos acceso a un remote proxy server desde una máquina remota (*remote login*) –no desde la máquina local–, se puede hacer un túnel para conectar el navegador a través del túnel en la máquina local. Consideremos que tenemos *login* sobre una maquina `gateway`, la cual puede acceder a la máquina llamada `proxy`, que ejecuta el Squid proxy server sobre el puerto 3128. Ejecutamos:

```
ssh -C -L 8080:proxy:3128 user@gateway
```

Después de conectarnos tendremos un túnel escuchando sobre el puerto local 8080, que reconducirá el tráfico desde gateway hacia proxy al 3128. Para navegar en forma segura, solo se deberá hacer http://localhost:8080/.

4. Servicios de transferencia de ficheros: FTP

El FTP (*file transfer protocol*) es un protocolo cliente/ servidor (bajo TCP) que permite la transferencia de archivos desde y hacia un sistema remoto. Un servidor FTP es un ordenador que ejecuta el *daemon* ftpd.

Algunos sitios que permiten la conexión anónima bajo el usuario *anonymous* son generalmente repositorios de software. En un sitio privado, se necesitará un usuario y un *password* para acceder. También es posible acceder a un servidor ftp mediante un navegador, y generalmente hoy en día los repositorios de software son sustituidos por servidores de web (p. ej. Apache) u otras tecnologías como Bittorrent (que utiliza redes PeerToPeer -P2P-). No obstante, se continúa utilizando en algunos casos y Debian, por ejemplo, acceso con usuario o passwd o la posibilidad de subir archivos al servidor (si bien con servicios web también es posible hacerlo). El protocolo (y servidores/clientes que lo implementan) de ftp por definición no son encriptados (los datos, usuarios y passwds se transmiten en texto claro por la red) con el riesgo que ello supone. Pero hay una serie de servidores/clientes que soportan SSL y por lo tanto encriptación.

4.1. Cliente ftp (convencional)

Un cliente ftp permite acceder a servidores FTP y hay una gran cantidad de clientes disponibles. La utilización del ftp es sumamente simple, desde la línea de comando, ejecutar:

ftp *nombre -servidor*

O también ftp, y luego en forma interactiva:

open *nombre -servidor*

El servidor solicitará un *username* y un *password* (si acepta usuario anónimos, se introducirá *anonymous* como usuario y nuestra dirección de e-mail como *password*) y a partir del prompt del comando (después de algunos mensajes), podremos comenzar a transferir ficheros.

El protocolo permite transferencia en modo ASCII o binarios. Es importante decidir el tipo de fichero que hay que transferir porque una transferencia de un binario en modo ASCII inutilizará el fichero. Para cambiar de un modo a

otro, se debe ejecutar el comando ascii o binary. Comandos útiles del cliente ftp son el ls (navegación en el directorio remoto), get nombre_del_fichero (para descargar ficheros) o mget (que admite *), put nombre_del_fichero (para enviar ficheros al servidor) o mput (que admite *); en estos dos últimos se debe tener permiso de escritura sobre el directorio del servidor. Se pueden ejecutar comandos locales si antes del comando se inserta un '!'. Por ejemplo !cd /tmp significará que los archivos que bajen a la máquina local se descargarán en /tmp. Para poder ver el estado y el funcionamiento de la transferencia, el cliente puede imprimir marcas, o ticks, que se activan con los comandos hash y tick. Existen otros comandos que se pueden consultar en la hoja del manual (man ftp) o haciendo *help* dentro del cliente.

Contamos con numerosas alternativas para los clientes, por ejemplo en modo texto: ncftp, lukemftp, lftp, cftp, yafc Yafc, o en modo gráfico: gFTP, WXftp, LLNL XFTP, guiftp. [Bor00]

4.2. Servidores FTP

El servidor tradicional de UNIX se ejecuta a través del puerto 21 y es puesto en marcha por el *daemon* inetd (o xinetd según se tenga instalado). En inetd.conf es conveniente incluir el wrapper tcpd con las reglas de acceso en host.allow y el host.deny en la llamada al ftpd por el inetd para incrementar la seguridad del sistema (consultar el capítulo dedicado a la seguridad). Cuando recibe una conexión, verifica el usuario y el *password* y lo deja entrar si la autentificación es correcta. Un FTP *anonymous* trabaja en forma diferente, ya que el usuario sólo podrá acceder a un directorio definido en el archivo de configuración y al árbol subyacente, pero no hacia arriba, por motivos de, seguridad. Este directorio generalmente contiene directorios pub/, bin/, etc/, y lib/ para que el *daemon* de ftp pueda ejecutar comandos externos para peticiones de ls. El *daemon* ftpd soporta los siguientes archivos para su configuración:

- /etc/ftpusers: lista de usuarios que no son aceptados sobre el sistema, un usuario por línea.

- /etc/ftpchroot: lista de usuarios a los que se les cambiará el directorio base chroot cuando se conecten. Necesario cuando deseamos configurar un servidor anónimo.

- /etc/ftpwelcome: anuncio de bienvenida.

- /etc/motd: noticias después del *login*.

- /etc/nologin: mensaje que se muestra después de negar la conexión.

- /var/log/ftpd: *log* de las transferencias.

Si queremos en algún momento inhibir la conexión al ftp, se puede hacer incluyendo el archivo /etc/nologin. El ftpd muestra su contenido y termina. Si existe un archivo .message en un directorio, el ftpd lo mostrará cuando se acceda al mismo.

La conexión de un usuario pasa por cinco niveles diferentes:

1) Tener una contraseña válida.

2) No aparecer en la lista de /etc/ftpusers.

3) Tener un *shell* estándar válido.

4) Si aparece en /etc/ftpchroot, se le cambiará al directorio home (incluido si es *anonymous* o ftp).

5) Si el usuario es *anonymous* o ftp, deberán tener una entrada en el /etc/passwd con user ftp, pero podrán conectarse especificando cualquier passwd (por convención se utiliza la dirección de e-mail).

Es importante tener en cuenta que los usuarios que sólo estén habilitados para utilizar el servicio ftp no dispongan de un *shell* a la entrada correspondiente de dicho usuario en /etc/passwd para impedir que este usuario tenga conexión, por ejemplo, por ssh o telnet. Para ello, cuando se cree el usuario, habrá que indicar, por ejemplo:

```
useradd -d/home/nteum -s /bin/false nteum
```

Y luego:

```
passwd nteum
```

Lo cual indicará que el usuario nteum no tendrá *shell* para una conexión interactiva (si el usuario ya existe, se puede editar el fichero /etc/passwd y cambiar el último campo por /bin/false). Luego se debe agregar como última línea /bin/false en /ect/shells. En [Mou01] se describe paso a paso cómo crear tanto un servidor ftp seguro con usuarios registrados como un servidor ftp *anonymous* para usuarios no registrados. Dos de los servidores no estándares más comunes son el WUFTPD (http://www.wuftpd.org) y el ProFTPD (http://www.proftpd.org). [Bor00, Mou01]

Para instalar el Proftpd sobre Debian ejecutar: `apt-get install proftpd`. Después de descargado, debconf le preguntará si lo quiere ejecutar por inetd o en modo manual (es recomendable elegir la última). Si se quiere parar el servicio (para cambiar la configuración por ejemplo), `/etc/init.d/proftpd stop`, y para modificar el fichero, /etc/proftpd.conf.

Consultar http://www.debian-administration.org/articles/228 para configurarlo en modo encriptado (TSL) o para tener acceso *anonymous*.

Un servidor (Debian) que es muy interesante es el PureFtpd (pure-ftpd) que es muy seguro, permite usuarios virtuales, cuotas, SSL/TSL, y un conjunto de características interesantes. Su instalación/configuración puede consultarse en http://www.debian-administration.org/articles/383.

5. Servicios de intercambio de información a nivel de usuario

5.1. El *mail transport agent* (MTA)

Un MTA (*mail trasnport agent*) se encarga de enviar/recibir los correos desde un servidor de e-mail hacia/desde Internet, que implementa el protocolo SMTP (*simple mail transfer protocol*). Debian utiliza por defecto exim, ya que es más fácil de configurar que otros paquetes MTA, como son smail o sendmail (este último es uno de los precursores). exim presenta características avanzadas tales como rechazar conexiones de sitios de SPAM conocidos, posee defensas contra junk mails o mail bombing y es extremadamente eficiente en el procesamiento de grandes cantidades de correos. Su ejecución se realiza a través de inetd en una línea en el archivo de configuración /etc/inetd.conf con parámetros para configuraciones normales (o xinetd).

exim utiliza un archivo de configuración en /etc/exim/exim.conf, que puede ser modificado manualmente, pero es recomendable hacerlo con un *shell script* llamado `eximconfig`, para poder configurar exim en forma interactiva. Los valores de la configuración dependerán de la situación de la máquina; sin embargo, su conexión es sumamente fácil, ya que el mismo *script* sugiere valores por defecto. No obstante, en /usr/doc/exim pueden encontrarse ejemplos de configuración típicas.

Se puede probar si la configuración es válida con exim -bV y, si hay errores en el archivo de configuración, el programa los mostrará por pantalla o, si todo está correcto, sólo pondrá la versión y fecha. Para probar si puede reconocer un buzón (*mailbox*) local, utilizad:

```
exim -v -bt usuario_local
```

Donde se mostrarán las capas de transporte utilizadas y la dirección local del usuario. También se puede hacer el siguiente test con un usuario remoto reemplazando usuario local por una dirección remota para ver su comportamiento. Luego intentad enviar un correo local y remotamente, pasando directamente los mensajes a exim (sin utilizar un agente por ejemplo, mailx), tecleando por ejemplo (todo junto):

```
exim postmaster@SuDominio
From: user@dominio
To: postmaster@SuDominio
Subject: Test Exim
Mensaje de prueba
^D
```

A continuación, se pueden analizar los archivos de traza mainlog y paniclog en /var/log/exim/ para ver su comportamiento y cuáles son los mensajes de error generados. Obviamente, también podéis conectaros al sistema como el usuario *postmaster* (o al cual se haya enviado el mail) y leer los correos para ver si todo es correcto. La otra forma consiste en ejecutarlo en modo *debug* utilizando como parámetro -dNro, donde Nro es el nivel de *debug* (19). El parámetro normal con el cual se debe poner en marcha es exim -bs, ya sea por inetd o por xinetd. También es posible ejecutarlo como *daemon* a través de /etc/init.d/exim start en sistemas que necesiten prestaciones elevadas al tratamiento de los correos. Consultar la documentación (incluida en Debian el paquete exim-doc-html) para configurar filtros, verificación de *hosts*, de *sender*, etc. Es interesante también instalar el paquete eximon, que es un monitor del exim y permite al administrador ver la cola de correos, logs y realizar diferentes acciones con los mensajes en cola para ser distribuidos (freezing, bouncing, thawing...).

La última versión de Exim es Exim4 (se puede instalar con `apt-get install exim4-daemon-heavy` (y también instalar exim4-config que servirá para configurar exim4) –tener en cuenta que hay diferentes paquetes con diferentes posibilidades pero exim4-daemon-heavy es la más completa). Es recomendable leer /usr/share/doc/exim/README.Debian.gz and update-exim4.conf(8). Para más información se puede consultar el HowTo http://www.exim.org/docs.html. Unas pequeñas diferencias a tener en cuenta en la configuración es que en lugar de tener una única configuración exim.conf (que es lo que tendrá si se instala exim desde los fuentes) el paquete exim4-config (es conveniente instalarlo para configurar exim4) utiliza pequeños archivos de configuración en lugar de uno único y que estarán en /etc/exim4/conf.d/* y serán concatenados todos en un único archivo (/var/lib/exim4/config.autogenerated por defecto) por update-exim4.conf.

5.2. *Internet message access protocol* (IMAP)

Este servicio permite acceder a los correos alojados en un servidor a través de un cliente de correo como por ejemplo Thunderbird o el cliente de correo de Seamonkey (ambos en mozilla.org). Este servicio soportado por el *daemon imapd* (los actuales soportan el protocolo IMAP4rev1) permite un archivo de correo electrónico (*mail file*) que se encuentra en una máquina remota. El servicio imapd se presta a través de los puertos 143 (*imap2*) o 993 (*imaps*) cuando soporta encriptación por SSL. Si se utiliza inetd, este servidor se pone en marcha a través de una línea en /etc/inetd.conf como:

```
imap2 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
imap3 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
```

En este ejemplo se llama al wrapper tcpd que funciona con hosts.allow y hosts.deny para incrementar la seguridad. Las aplicaciones más populares son

uw-imapd (Universidad de Washington e instalado por defecto en Debian) o su versión segura uw-imapd-ssl, cyrus-imap o courier-imap. Para probar que el servidor imap funciona, se podría utilizar un cliente, por ejemplo, `seamonkey -mail` y crear una cuenta para un usuario local y configurarlo adecuadamente para que se conecte sobre la máquina local, verificando el funcionamiento de imap.

Sobre Debian, la versión de imap ha sido compilada para soportar MD5 como método de autentificación de los usuarios remotos, para encriptar los *passwords* de conexión y evitar suplantación de identidad por *sniffing* en la red (el cliente utilizado para conectarse al servidor imap también debe soportar el método de autentificación por MD5). El método es muy simple y seguro, pero el servidor debe conocer los *passwords* en texto plano de los usuarios de correo, por lo cual se recomienda utilizar la versión de imapd sobre SSL y que funciona sobre el puerto 993. El protocolo imaps que al igual que ssh se basa en encriptar la comunicación a través de un certificado del *host* (el cliente utilizado para conectarse al servidor también debe soportar este método de conexión por ejemplo, thunderbird o seamonkey -mail). Para configurar el servidor imaps, instalad el paquete uw-imap-dssl de Debian que es el servidor imap con soporte SSL.

La instalación genera un certificado autofirmado válido por un año y lo almacena en /etc/ssl/certs/imapd.pem. Este certificado se puede reemplazar por uno firmado por una compañía certificadora o se puede generar uno propio con OpenSSL. Es conveniente dejar sólo la entrada imaps en el archivo /etc/inetd.conf y quitar las entradas imap2 e imap3 si únicamente se quiere que el acceso a imap sea por SSL.

Otro protocolo de similares características que ha tenido mucha popularidad en el pasado, pero que hoy se ha visto superado por IMAP, es POP (*post office protocol*) versión 2 y 3. Su instalación y puesta en marcha es análoga a la de IMAP. Existen multitud de servidores POP, pero los más comunes son courier-pop, cyrus-pop3d, ipopd (Universidad de Washington), qpopper, solid-pop3d.

5.2.1. Aspectos complementarios

Supongamos que como usuarios tenemos 4 cuentas de correos en servidores diferentes y queremos que todos los mails que llegan a estas cuentas se recojan en una única, que podamos acceder externamente a esta cuenta y que haya un filtro de correo basura (*antispam*) también.

Primero se debe instalar Exim + Imap y comprobar que funcionan. Se debe tener en cuenta que si se instala courier-imap (que según algunos autores es mejor que uw-imapd) éste funciona sobre un formato de mail llamado Maildir, que se debería configurar Exim para que también funcione sobre maildir con

la siguiente configuración en /etc/exim/exim.conf (o en la correspondiente si se tiene exim4), cambiando la opción mail_dir format = true (los correos se guardarán en la cuenta del usuario local en un directorio llamado Maildir). Luego se debe reiniciar el servidor exim con `/etc/init.d/exim restart`, repetir la prueba de funcionamiento enviándonos un mail y leerlo con un cliente que soporte maildir (por ejemplo `mutt -mailx` no lo soporta- ver http://www.mutt.org).

Para recoger los mails de diferentes cuentas se utilizará fetchmail, (que se instala con `apt-get install fetchmail`). Se debe a continuación crear el fichero .fetchmailrc en nuestro $HOME (también se puede utilizar la herramienta fetchmailconf) que deberá tener ser algo así como:

```
set postmaster "pirulo"
set bouncemail
set no spambounce
set flush

poll pop.domain.com proto pop3
user 'user1' there with password 'secret' is pirulo here

poll mail.domain2.com
 user 'user5' there with password 'secret2' is 'pirulo' here
 user 'user7' there with password 'secret3' is 'pirulo' here
```

La acción `set` indica a Fetchmail que esta línea contiene una opción global (envío de errores, borrar los mail de los servidores...). A continuación, se especifican dos servidores de correo: uno para que compruebe si hay correo con el protocolo POP3 y otro para que pruebe a usar varios protocolos para encontrar uno que funcione. Se comprueba el correo de dos usuarios con la segunda opción de servidor, pero todo el correo que se encuentre se envía al spool de correo del pirulo. Esto permite comprobar varios buzones de diversos servidores como si se tratara de un único buzón MUA. La información específica de cada usuario comienza con la acción `user`. EL fetchmail se puede poner en el cron (por ejemplo en /var/spool/cron/crontabs/pirulo agregando 1 * * * * /usr/bin/fetchmail -s), para que se ejecute automáticamente o ejecutarlos en modo *daemon* (poner set daemon 60 en .fetchmailrc y ejecutarlo una vez por ejemplo en Autostart de Gnome/KDE o en el .bashrc -se ejecutará cada 60 segundos).

Para quitar el correo basura se usará SpamAssassin (apt-get install spamassassin) y se puede configurar Kmail o Evolution (ver bibliografía para consultar como configurarlo) para que lo ejecuten. En esta configuración se utilizará Procmail que es una herramienta muy potente (permite repartir el correo, filtrarlo, reenviarlo automáticamente...). Una vez instalado (apt-get install procmail), se debe crear un fichero llamado .procmailrc en el home de cada usuario que llamará al Spamassassin:

- Poner *yes* para mensajes de funcionamiento o depuración

```
VERBOSE=no
```

- Consideramos que los mails están en "~/.Maildir"), cambiar si es otro PATH=/usr/bin:/bin:/usr/local/bin:.

  MAILDIR=$HOME/Maildir
  DEFAULT=$MAILDIR/
  – Directorio para almacenar los ficheros
    PMDIR=$HOME/.procmail
  – Comentar si no queremos *log* de Procmail
    LOGFILE=$PMDIR/log
  – filtro de Smap
    INCLUDERC=$PMDIR/spam.rc

El archivo ~/.procmail/spam.rc contiene:

- Si el spamassassin no está en el PATH agregar a la variable PATH el directorio:
  0fw: spamassassin.lock
  | spamassassin -a

- La tres líneas siguientes moverán el correo Spam a un directorio llamado "spam-folder" Si se quiere guardarlo en el Inbox, para luego filtrarlo con el cliente, comentar las tres líneas.

  :0:
  * ^X-Spam-Status: Yes
  spam-folder

El archivo ~/.spamassassin/user_prefs contiene alguna configuraciones útiles para spamassassin (consultar la bibliografía):

- user preferences file. Ver man Mail::SpamAssassin::Conf

- Umbral para reconocer un Spam: Default 5, pero con 4 funciona un poco mejor
  required_hits 4

- Sitios que nunca consideraremos que vendrá Spam
  whitelist_from root@debian.org
  whitelist_from *@uoc.edu

- Sitios que siempre viene SPAM (separado por comas)
  blacklist_from viagra@dominio.com

- direcciones en Whitelist y blacklist son patrones globales como: "amigo@lugar.com", "*@isp.net", o "*.domain.com" .

- Insertar la palabra "[SPAM]" en el subject (facilita hacer filtros).

- Si no se desea comentar la línea.
  subject_tag [SPAM]

Esto generará un tag X-Spam-Status: *Yes* en la cabecera del mensaje si cree que el mensaje es Spam. Luego se deberán filtrar éstos y ponerlos en otra carpeta o borrarlos directamente. Se puede utilizar el procmail para filtrar mails de dominios, usuarios etc. Para más información consultar http://www.debian-administration.org/articles/242. Por último se puede instalar un cliente de correo y configurar los filtros para que seleccione todos los correos con X-Spam-Status: *Yes* y los borre o los envíe a un directorio que luego verificaremos los falsos positivos (correos identificados como basura pero que no lo son). Un aspecto complementario en esta instalación es si se desea tener un servidor de correo a través de webmail (es decir, poder consultar los correos del servidor a través de una navegador sin tener que instalar un cliente ni configurarlo –igual que consultar una cuenta de gmail o hotmail) es posible instalar Squirrelmail (`apt-get install squirrelmail`) para dar este servicio. Para Debian consultar http://www.debian-administration.org/articles/200.

Hay otra posibilidades como se comenta en http://www.debian-administration.org/articles/364 instalando MailDrop en lugar de Procmail, Postfix el lugar de Exim, o incluyendo Clamav/Amavisd como antivirus (Amavisd permite vincular postfix con spamassassin y clamav).

5.3. *News*

Las *news* o grupos de discusión son soportados a través del protocolo NNTP. Instalar un servidor de *news* es necesario cuando se desea leer *news* fuera de línea, cuando se quiere tener un repetidor de los servidores centrales o se quiere un propio servidor master de *news*. Los servidores más comunes son INN o CNEWS, pero son paquetes complejos y destinados a grandes servidores. Leafnode es un paquete USENET que implementa servidor TNP, especialmente indicado para sitios con grupos reducidos de usuarios, pero donde se desea acceder a gran cantidad de grupos de noticias. Este servidor se instala en la configuración básica de Debian y se puede reconfigurar con `dpkg-reconfigure leafnode`, todos parámetros como los servidores centrales, el tipo de conexión, etc. Este *daemon* se pone en marcha desde inetd en forma similar al imap (o con xinetd). Leafnode soporta filtros a través de expresiones regulares indicadas (del tipo ^Newsgroups:. * [,] alt.flame$) en /etc/news/leafnode/filters, donde para cada mensaje se compara la cabecera con la expresión regular y, si existe coincidencia, el mensaje es rechazado.

La configuración de este servidor es simple y todos los archivos deben ser propiedad de un usuario *news* y con permiso de escritura (verificar que dicho propietario existe en /etc/passwd). Todos los archivos de control, *news* y configuración se encuentran en /var/spool/news excepto la configura-

ción del propio servidor que está en el fichero /etc/news/leafnode/config. En la configuración existen algunos parámetros obligatorios que deben ser configurados (por ejemplo, para que el servidor pueda conectarse con los servidores maestros). Ellos son *server* (servidor de *news* desde donde se obtendrán y enviarán las *news*) y *expire* (número de días que un *thread* o sesión ha sido leída y se borrará). Tenemos, asimismo, un conjunto de parámetros opcionales de ámbito general o específicos del servidor que podrían configurarse. Para más información, consultad la documentación (*man leafnode* o /usr/doc/leafnode/README.Debian).

Para verificar el funcionamiento del servidor, se puede hacer:

```
telnet localhost nntp
```

y si todo funciona correctamente, saldrá la identificación del servidor y se quedará esperando un comando, como prueba, se puede introducir *help* (para abortar, hacer Ctrl+ (y luego Quit).

5.4. World Wide Web (httpd)

Apache es uno de los servidores más populares y de altas prestaciones de HTTP (*hypertext transfer protocol*). Apache tiene un diseño modular y soporta extensiones dinámicas de módulos durante su ejecución. Es altamente configurable en el número de servidores y de módulos disponibles y soporta diversos mecanismos de autentificación, control de acceso, *metafiles*, *proxy caching*, servidores virtuales, etc. Con módulos (incluidos en Debian) es posible tener PHP3, Perl, Java Servlets, SSL y otras extensiones (podéis consultar la documentación en http://www.apache.org).

Apache está diseñado para ejecutarse como un proceso *daemon standalone*. En esta forma crea un conjunto de procesos hijo que manejarán las peticiones de entrada. También puede ejecutarse como Internet *daemon* a través de inetd, por lo cual se pondrá en marcha cada vez que se reciba una petición. La configuración del servidor puede ser extremadamente compleja según las necesidades (consultad la documentación), sin embargo, aquí veremos una configuración mínima aceptable. Los archivos de configuración se encuentran en /etc/apache y son httpd.conf (archivo principal de configuración), srm.conf, access.conf (estos dos últimos son mantenidos por compatibilidad y su funcionalidad está en el anterior), mime.conf (formatos MIME) y magic (número de identificación de archivos). Los archivos *log* se encuentran en /var/log/apache y son error.log (registra los errores en las peticiones del servidor), access.log (registro de quién y a qué ha accedido) y apache.pid (identificador del proceso).

Apache se pone en marcha desde el *script* de inicio /etc/init.d/apache y los /etc/rcX.d, pero puede controlarse manualmente mediante el comando `apachectl`. También se puede utilizar el comando apacheconfig para configurar el servidor. Los directorios por defecto (en Debian) son:

- /var/www: directorio de documentos HTML.
- /usr/lib/cgibin: directorio de ejecutables (*cgi*) por el servidor.
- http://server.dominio/ user: páginas personales de los usuarios.
- /home/user/public.html: directorio de páginas personales.

El archivo por defecto que se lee de cada directorio es index.html. Una vez instalados los paquetes *apache y apache-common*, Debian configura básicamente el servidor y lo pone en marcha. Se puede comprobar que funciona abriendo un *browser* (por ejemplo, el Konqueror, y poniendo en la barra de URL http://localhost, lo cual cargará la página /var/www/index.html).

5.4.1. Configuración manual (mínima) de httpd.conf

Vamos a ver algunos de los parámetros más importantes en la configuración de Apache (el ejemplo está tomado de la versión 1.X de Apache y existen algunos cambios menores si se utiliza la versión 2).

ServerType standalone	Recomendado, más eficiente
ServerRoot /etc/apache	Donde están los archivos de configuración
Port 80	Donde el servidor escuchará las peticiones
User www-data	*User* y *group* con los cuales se ejecutará el servidor (importante por seguridad) deben ser usuarios válidos (pueden estar *locked*)
Group www-data	
ServerAdmin webmaster@pirulo.remix.com	Dirección de usuario que atenderá los errores
ServerName pirulo.remix.com	Nombre del servidor enviado a los usuarios –debe ser un nombre válido en /etc/host o DNS–
DocumentRoot /var/www	Directorio donde estarán los documentos
Alias /icons/ /usr/share/apache/icons/	Donde se encuentran los iconos
ScriptAlias /cgibin/ /usr/lib/cgibin/	Donde se encuentran los *script* CGI

5.4.2. Apache 2.2 + SSL + PHP + MySQL

Un aspecto importante para servidores web dinámicos es aprovechar las ventajas de Apache en modo seguro (SSL), PHP (es un lenguaje de programación usado generalmente para la creación de contenido para sitios web)

y MySQL+PHPAdmin (base de datos que hablaremos en próximos capítulos e interfaz gráfica para su gestión) todo ello funcionando conjuntamente. Partiremos de la base de instalarlo sobre un Debian Sarge, pero no a través de paquetes deb sino desde el software bajado de los sitios respectivos, así se puede repetir la experiencia sobre otras distribuciones. Obviamente estos paquetes después no podrán ser controlados por apt u otro gestor de paquetes. Se debe tener cuidado con las versiones que pueden cambiar y de no superponer la instalación a paquetes ya instalados.

a) Descarga de los ficheros necesarios (por ejemplo dentro del directorio /root -> cd /root):

1) Apache: desde http://httpd.apache.org/download.cgi: httpd-2.2.4.tar.bz2

2) PHP: desde http://www.php.net/downloads.php PHP 5.2.1 (tar.bz2)

3) MySQL desde http://mysql.org/get/Downloads/MySQL-4.1/mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz/from/pick

4) PHPAdmin desde http://prdownloads.sourceforge.net/phpmyadmin/phpMyAdmin-2.9.1-all-languages.tar.bz2?download

b) Utilidades: bzip2 libssl-dev openssl gcc g++ cpp make (verificar que no se tienen instaladas o si no, hacer apt-get install bzip2 libssl-dev openssl gcc g++ cpp make.

c) Apache:

```
cd /root
tar jxvf httpd-2.2.4.tar.bz2
cd httpd-2.2.4
```

Con prefix, indicamos se instalará por ejemplo `/usr/local/apache2`

```
./configure --prefix=/usr/local/apache2 –with ssl=/usr/include/openssl \
--enable-ssl
make
make install
```

Modificamos el fichero de configuración `/usr/local/apache2/conf/httpd.conf` y cambiamos el usuario y grupo de trabajo por www-data:

```
User www-data
Group www-data
```

Cambiamos el dueño y grupo del directorio de datos a www-data:

```
chown -R www-data:www-data /usr/local/apache2/htdocs
```

Modificamos el usuario www-data para cambiar su directorio *home* en /etc/passwd:

```
www-data:x:33:33:www-data:/usr/local/apache2/htdocs:/bin/sh
```

Servidor apache instalado. Para iniciarlo (para pararlo cambiar *start* por *stop*):

```
/usr/local/apache2/bin/apachectl start
```

Se puede colocar un *script* para arrancar el servidor apache al *boot*.

```
ln -s /usr/local/apache2/bin/apachectl /etc/rcS.d/S99apache
chmod 755 /etc/rcS.d/S99apache
```

d) SSL:

En /usr/local/apache2/conf/httpd.conf quitamos el comentario de la línea:

```
Include conf/extra/httpd-ssl.conf
```

Se generan los ficheros con las claves para el servidor seguro, en /root hacemos (adecuar las versiones a las que se hayan descargado) –el primer comando openssl es una línea entera y acaba con 1024:

```
openssl genrsa -rand ../httpd-2.2.4.tar.bz2:../php-5.2.1.tar.bz2:../phpMyAdmin-2.9.1-all-languages.tar.bz2 -out server.key 1024
openssl rsa -in server.key -out server.pem
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 720 -in server.csr -signkey server.key -out server.crt
```

Se copian los ficheros...

```
cp server.crt /usr/local/apache2/conf/
cp server.key /usr/local/apache2/conf/
```

Reiniciamos el servidor...

```
/usr/local/apache2/bin/apachectl restart
```

Se puede consultar cómo agregar el módulo SSL a un servidor que no lo tenga instalado en http://www.debian-administration.org/articles/349.

e) MySQL (para más información ver el módulo 8):

Creamos un grupo y un usuario para MySQL si no existe

```
groupadd mysql
useradd -g mysql mysql
```

En el directorio donde se instalará MySQL (/usr/local/) hacemos

```
cd /usr/local/
gunzip < /root/mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz | tar xvf -
ln -s mysql-standard-4.1.21-pc-linux-gnu-i686 mysql
cd mysql
```

Creo una base de datos y cambio los permisos

```
scripts/mysql_install_db --user=mysql
chown -R root.
chown -R mysql data
chgrp -R mysql.
```

Se puede colocar un *script* para iniciar el servidor mysql.

```
ln -s /usr/local/mysql/support-files/mysql.server /etc/rcS.d/S99mysql.server
chmod 755 /etc/rcS.d/S99mysql.server
```

Iniciamos el servidor

```
/etc/rcS.d/S99mysql.server start
```

Se puede entrar en la BD y cambiar el *password* del *root* por seguridad (consultar http://dev.mysql.com/doc/refman/5.0/en/index.html para la sintaxis)

```
/usr/local/mysql/bin/mysql
```

Dentro hacemos:

USE mysql

Colocamos el *password* pirulo al usuario *root*

UPDATE user SET Password=PASSWORD('pirulo') WHERE User='root';

FLUSH privileges;

Para entrar en MySQL deberemos hacer

```
/usr/local/mysql/bin/mysql -u root -ppirulo
```

f) PHP (reemplazar con las versiones adecuadas):

Utilidades necesarias:

```
apt-get install libxml2-dev curl libcurl3-dev libjpeg-mmx-dev zlib1g-dev \
libpng12-dev
```

Con el servidor Apache parado hacemos:

```
cd /root
tar jxvf php-5.2.0.tar.bz2
cd php-5.2.0
```

Con prefix se puede indicar dónde se quiere instalar (todo en una línea):

```
./configure --prefix=/usr/local/php5 --enable-mbstring --with-apxs2=/
usr/local/apache2/bin/apxs --with-mysql=/usr/local/mysql --with-curl=/
usr/include/curl --with-jpeg-dir=/usr/include --with-zlib-dir=/usr/
include --with-gd --with-xml --enable-ftp --enable-bcmath

make
make install
cp php.ini-dist /usr/local/php5/lib/php.ini
```

Modificamos Apache (/usr/local/apache2/conf/httpd.conf) en la parte indicada:

```
<IfModule mime_module>
    AddType application/x-httpd-php .php .phtml
    AddType application/x-httpd-php-source .phps
```

Y también:

DirectoryIndex index.php index.html

Reiniciamos el servidor.

g) PHPAdmin

```
cd /usr/local/apache2/
```

Se descomprime phpmyadmin en el directorio de apache2 (cuidado con las versiones).

```
tar jxvf /root/phpMyAdmin-2.9.1-all-languages.tar.bz2
mv phpMyAdmin-2.9.1-all-languages phpmyadmin
cd phpmyadmin
cp config.sample.inc.php config.inc.php
```

Se debe modificar el fichero de configuración (config.inc.php):

$cfg['blowfish_secret'] = 'pirulo';

Quito el usuario y *password* del usuario por defecto dos (') seguidas:

```
$cfg['Servers'][$i]['controluser'] = '';
$cfg['Servers'][$i]['controlpass'] = '';
```

Cambio apache (/usr/local/apache2/conf/httpd.conf) añadiendo en <IfModule alias_module>

```
<IfModule alias_module>
        Alias /phpmyadmin "/usr/local/apache2/phpmyadmin/"
<Directory "/usr/local/apache2/phpmyadmin/">
        Order allow,deny
        Allow from all
</Directory>
```

Reiniciamos el servidor y se puede llamar con http://localhost/phpadmin

Se puede tener más información en las webs respectivas de cada aplicación y en LWP.

6. Servicio de Proxy: Squid

Un servidor Proxy (PS) se utiliza para salvar ancho de banda de la conexión de red, mejorar la seguridad e incrementar la velocidad para obtener páginas de la Red (*web-surfing*).

Squid es uno de los principales PS, ya que es OpenSource, acepta ICP (características que le permiten intercambiar *hints* con otros PS), SSL (para conexiones seguras entre *proxies*) y soporta objetos FTP, Gopher, HTTP y HTTPS (seguro). Su funcionamiento es simple, almacena los objetos más solicitados en memoria RAM y los menos en una base de datos en el disco. Los servidores Squid, además, pueden configurarse de forma jerárquica para formar un árbol de *proxies* dependiendo de las necesidades. Existen dos configuraciones posibles:

1) Como acelerador de httpd para lograr más prestaciones al servicio de web.

2) Como *proxy-caching server* para permitir a los usuarios de una corporación utilizar el PS para salir hacia Internet.

En el primer modo, actúa como *proxy* inverso, es decir, acepta una petición del cliente, sirve el objeto si lo tiene y si no, lo solicita y se lo pasa al cliente cuando lo tiene, almacenándolo para la vez siguiente. En la segunda opción se puede utilizar como control y para restringir los sitios donde se puede conectar en Internet o autorizar el acceso a determinadas horas del día. Una vez instalado (paquete squid en Debian, también se puede instalar squid-cgi, squidguard o squidtaild) se generan tres archivos: /etc/squid.conf (configuración), /etc/init.d/squid (inicialización) y /etc/logrotate.d/squid (de control de los logs).

6.1. Squid como acelerador de http

En este modo, si el servidor de web está en la misma máquina donde está el PS, se deberá reconfigurar para que atienda peticiones del puerto 81 (en Apache, cambiar Port 80 por Port 81 en httpd.conf). El archivo de configuración (/etc/squid.conf) contiene una gran cantidad de entradas, pero aquí solo veremos las indispensables [Mou01]:

```
http_port 80                 Donde escucha httpd
icp_port 0                   Donde escucha ICP
```

hierarchy_stoplist cgi-bin \?	
acl QUERY urlpath_regex cgi-bin \?	
no_cache deny QUERY	
cache_mem 100 MB	Memoria para objetos en curso
redirect_rewrites_host_header off	
cache_replacement_policy lru	
memory_replacement_policy lru	
cache_dir ufs /var/spool/squid 100 16 256	Tipo y lugar donde está la Base de Datos de caché de disco
emulate_httpd_log on	
acl all src 0.0.0.0/0.0.0.0	Acceso para todos
http_access allow all	Y a todo
cache_mgr root	Mail responsable
cache_effective_user proxy	UID
cache_effective_group proxy	GID
httpd_accel_host 192.168.1.1	Servidor real de Web
httpd_accel_port 81	Puerto
logfile_rotate 0	
log_icp_queries off	
buffered_logs on	

De este modo, la opción httpd_accel_host desactiva la posibilidad de que se ejecute como proxy-caching. Para más información consultar http://www.squid-cache.org/.

6.2. Squid como *proxy-caching*

De esta manera se habilita el squid para que controle el acceso a Internet, cuándo accederán y a qué accederán. En este caso, el archivo de configuración deberá incluir las siguientes modificaciones/ agregados en /etc/squid.conf:

```
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 70 21 102565535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
http_access allow localnet
http_access allow localhost
http_access deny
http_access deny CONNECT
http_access deny all
cache_emulate_httpd_log on
```

La gran diferencia con el otro modo son las líneas acl, en cuyo caso se permitirá a los clientes de la clase C 192.168.1.0 acceder al PS, también el *localhost* IP y otros puertos que podrán acceder a Internet 80(http), 443(https), 210(whais), 70(gopher), and 21(ftp), además, se niega el método *connect* para evitar que desde fuera se puedan conectar al PS y luego se niegan todos los IP y puertos sobre el PS. [Mou01] Más información en http://www.squid-cache.org/ y para un transparent-proxy en http://tldp.org/HOWTO/TransparentProxy-1.html.

7. OpenLdap (Ldap)

LDAP significa *lightweight directory access protocol* y es un protocolo para acceder a datos basados en un servicio X.500. Éste se ejecuta sobre TCP/IP y el directorio es similar a una base de datos que contiene información basada en atributos. El sistema permite organizar esta información de manera segura y utilizando réplicas para mantener su disponibilidad, asegurando la coherencia y la verificación sobre los datos accedidos-modificados.

El servicio se basa en el modelo cliente-servidor, donde existe un servidor o más de uno que contiene los datos; cuando un cliente se conecta y solicita información, el servidor responde con los datos o un puntero a otro servidor donde podrá extraer más información, pero el cliente sólo verá un directorio de información global. [Mou01, Mal07]

Para importar y exportar información entre servidores ldap, o para describir una serie de cambios que serán aplicados al directorio, el formato utilizado se llama LDIF (LDAP *data interchange format*). LDIF almacena la información en jerarquías orientadas a objetos que luego serán transformadas al formato interno de la base de datos. Un archivo LDIF tiene un formato similar a:

```
dn: o = UOC, c = SP
o: UOC
objectclass: organization
dn: cn = Pirulo Nteum, o = UOC, c = SP
cn: Pirulo Nteum
sn: Nteum
mail: nteum@uoc.edu
objectclass: person
```

Cada entrada es identificada por un nombre indicado como DN (*distinguished name*). El DN consiste en el nombre de la entrada más una serie de nombres que lo relacionan con la jerarquía del directorio y donde existe un *objectclass* que define los atributos que pueden ser utilizados en esta entrada. LDAP provee un conjunto básico de clases de objetos: grupos (incluye listas desordenadas de objetos individuales o grupos de objetos), localizaciones (tales como países y su descripción), organizaciones y personas. Una entrada puede, además, pertenecer a más de una clase de objeto, por ejemplo, un individuo es definido por la clase persona, pero también puede ser definido por atributos de las clases inetOrgPerson, groupOfNames, y organization. La estructura de objetos del servidor (llamado *schema*) determina cuáles son los atributos permitidos para un objeto de una clase (los cuales se definen en /etc/ldap/schema como opeldap.schema, corba.schema, nis.schema, inetorgperson. schema, etc.).

Todos los datos son representados como un par *atributo* = *valor* donde atributo es descriptivo de la información que contiene, por ejemplo, el atributo utilizado para almacenar el nombre de una persona es commonName, o cn, es decir, para una persona llamada Pirulo Nteum, será representado por cn: Pirulo Nteum y llevará asociado otros atributos de la clase persona como givenname: Pirulo surname: Nteum mail: pirulo@uoc.edu. En las clases existen atributos obligatorios y optativos y cada atributo tiene una sintaxis asociada que indica qué tipo de información contiene el atributo, por ejemplo, bin (*binary*), ces (*case exact string*, debe buscarse igual), cis (*case ignore string*, puede ignorarse M-m durante la búsqueda), tel (*telephone number string*, se ignoran espacios y '-'), dn (*distinguished name*). Un ejemplo de un archivo en formato LDIF podría ser:

```
dn:             dc = UOC, dc = com
objectclass:    top
objectclass:    organizationalUnit

dn:             ou = groups, dc = UOC, dc = com
objectclass:    top
objectclass:    organizationalUnit
ou:             groups

dn:             ou = people, dc = UOC, dc = com
objectclass:    top
objectclass:    organizationalUnit
ou:             people

dn:             cn = Pirulo Nteum, ou = people, dc = UOC, dc = com
cn:             Pirulo Nteum
sn:             Nteum
objectclass:    top
objectclass:    person
objectclass:    posixAccount
objectclass:    shadowAccount
                uid:pirulo
                userpassword:{crypt}p1pss2ii(0pgbs*do&@ = )eksd
                uidnumber:104
                gidnumber:100
                gecos:Pirulo Nteum
                loginShell:/bin/bash
                homeDirectory: /home/pirulo
                shadowLastChange:10877
                shadowMin: 0
                shadowMax: 999999
                shadowWarning: 7
                shadowInactive: -1
                shadowExpire: -1
                shadowFlag: 0

dn:             cn = unixgroup, ou = groups, dc = UOC, dc = com
objectclass:    top
objectclass:    posixGroup
cn:             unixgroup
                gidnumber: 200
                memberuid: pirulo
                memberuid: otro-usuario
```

Las líneas largas pueden ser continuadas debajo comenzando por un espacio o un tab (formato LDIF). En este caso, se ha definido la base DN para la institución dc = UOC, dc = com, la cual contiene dos subunidades: *people* y *groups*. Luego se ha descrito un usuario que pertenece a *people* y a *group*. Una vez pre-

parado el archivo con los datos, debe ser importado al servidor para que esté disponible para los clientes LDAP. Existen herramientas para transferir datos de diferentes bases de datos a formato LDIF. [Mal07]

Sobre Debian, se debe instalar el paquete slapd que es el servidor de OpenLdap. Durante la instalación realizará una serie de preguntas como: *Método de instalación del directorio:* auto; *extensiones al directorio [domain-host,sitio,institución]: host, domain, password del Adm*; *replicar cambios locales a otros servidores:* no. Esta instalación generará un archivo de configuración en /etc/ldap/slapd.conf y la base de datos sobre /var/lib/ldap. También existe otro archivo /etc/ldap/ldap.conf (o puede existir el ~/.ldaprc), que es el archivo de configuración utilizado para inicializar valores por defecto cuando se ejecutan clientes ldap. En éste se indica cuál es la base de datos, cuál es el servidor ldap, parámetros de seguridad, tamaño de la búsqueda, etc.

El archivo de configuración del servidor /etc/ldap/slapd.conf (ver `man slap.con`) está compuesto por diferentes secciones, cada una de ellas indicada por una de las siguientes directivas: global, backend specific y database specific, y en ese orden. La directiva *global* es de carácter general y se aplica a todos los *backends* (bases de datos) y definen cuestiones generales tales como los permisos de acceso, atributos, tiempos de espera, *schemas*, etc. La directiva *backend specific* define los atributos al *backend* específico que define (bdb, dnssrv, ldbm...), y el *database specific* los atributos específicos para esa base de datos que define. Para poner en marcha el servidor, se debe ejecutar:

```
/etc/init.d/slapd start (o stop para pararlo)
```

El sistema durante la instalación habrá creado los enlaces adecuados para ejecutarlo después del inicio.

7.1. Creación y mantenimiento de la base de datos

Existen dos métodos para insertar datos en la base de datos de LDAP. El primero es fácil y adecuado para pequeñas cantidades de datos, es interactivo y se deben utilizar herramientas tales como ldapadd (o cualquier otra como Ldap Browser http://www.iit.edu/~gawojar/ldap/) para insertar nuevas entradas. El segundo se debe trabajar fuera de línea, es el adecuado para grandes BD y se utiliza el comando slapadd incluido con slapd. Por ser más general, describiremos sintéticamente el segundo método, donde primero se debe verificar que contiene los siguientes atributos en slapd.conf: suffix (top del directorio, por ejemplo, suffix “o = UOC, c = SP”); directory /var/lib/ldap (directorio donde se crearán los índices y que pueda escribir slapd). Se debe además verificar que la base de datos contiene las definiciones de los índices que se desean:

```
index cn,sn,uid
index objectClass pres,eq
```

Una vez definido el slapd.conf, se debe ejecutar el comando:

```
slapadd -l entrada -f configuración [-d nivel] [-n entero | -b sufijo]
```

Los argumentos son:

-l: archivo en formato LDFI.
-f: archivo de configuración del servidor, donde se indican cómo crear los índices.
-d: nivel de depuración.
-n: Nro de base de datos, si se tiene más de una.
-b: especifica qué base de datos hay que modificar.

Existen otros comandos con slapd tales como slapindex, que permite regenerar los índices, y slapcat, que permite volcar la BD a un archivo en formato LDIF.

8. Servicios de archivos (NFS)

El sistema NFS permite a un servidor exportar un sistema de archivo para que puedan ser utilizados en forma interactiva desde un cliente. El servicio se compone de un servidor nfsd y un cliente (*mountd*) que permiten compartir un sistema de archivo (o parte de él) a través de la red.

En Debian instalar para el cliente `apt-get install nfs-common portmap` mientras que el server necesita `apt-get install nfs-kernel-server nfs-common portmap`.

El servidor (en Debian) se pone en marcha a través de los scripts nfscommon y nfs-kernel-server en /etc/init.d (y los enlaces adecuados en /etc/rcX.d).

El servidor utiliza un archivo (/etc/exports) para gestionar el acceso y control sobre los sistemas de archivo que se accederán remotamente. Sobre el cliente, el *root* (u otro usuario a través de *sudo*) puede montar el sistema remoto a través del comando:

```
mount Ipserver:directorio-remoto directorio_local
```

y a partir de este momento el directorio-remoto se verá dentro de directorio local (éste debe existir antes de ejecutar el *mount*). Esta tarea en el cliente se puede automatizar utilizando el archivo de *mount* automático (/etc/fstab) incluyendo una línea; por ejemplo:

pirulo.remix.com:/usr/local /pub nfs rsize=8192,wzise=8192,timeo=14

Esta sentencia indica que se montará el directorio /usr/local del host pirulo.remix.com en el directorio local /pub. Los parámetros rsize, wzise son los tamaños de bloques de lectura y escritura, timeo es el timeout de RPC (si no se especifican estos tres valores, se toman los que incluye por defecto).

El archivo /etc/exports sirve de ACL (lista de control de acceso) de los sistemas de archivo que pueden ser exportados a los clientes. Cada línea contiene un *filesystem* por exportar seguido de los clientes que lo pueden montar, separados por espacios en blanco. A cada cliente se le puede asociar un conjunto de opciones para modificar el comportamiento (consultar *man exports* para un lista detallada de las opciones). Un ejemplo de esto podría ser:

```
# Ejemplo de /etc/exports
/               /master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/usr            *.local.domain(ro) @trusted(rw)
/pub            (ro,insecure,all_squash)
/home           195.12.32.2(rw,no_root_squash) www.first.com(ro)
/user           195.12.32.2/24(ro,insecure)
```

La primera línea exporta el sistema de archivos entero (/) a master y trusty en modo lectura/escritura. Además, para trusty no hay *uid squashing* (el *root* del cliente accederá como *root* a los archivos *root* del servidor, es decir, los dos *root* son equivalentes a pesar de ser de máquinas diferentes; es indicado para máquinas sin disco). La segunda y tercera líneas muestra ejemplo de '*' y de *netgroups* (indicados por @). La cuarta línea exporta el directorio /pub a cualquier máquina en el mundo, sólo de lectura, permite el acceso de clientes NFS que no utilizan un puerto reservado para el NFS (opción *insecure*) y todo se ejecuta bajo el usuario *nobody* (opción *all squash*). La quinta línea especifica un cliente por su IP y en la sexta igual pero con máscara de red (/24) y con opciones entre () y que deben estar sin espacio de separación. Sólo puede haber espacios entre los clientes habilitados. Es importante tener en cuenta que de NFS existen 3 versiones (V2, V3 y recientemente V4). Las más comunes son V3 y en algunas instalaciones V2. Si desde un cliente V3 se conecta a un servidor V2, se debe indicar con un parámetro esta situación.

8.1. Servidor de Wiki

Un (o una) wiki (del hawaiano *wiki wiki*, "rápido") es un sitio web colaborativo que puede ser editado por varios usuarios que pueden crear, editar, borrar o modificar el contenido de una página web, de una forma interactiva, fácil y rápida; dichas facilidades hacen de una wiki una herramienta efectiva para la escritura colaborativa. La tecnología wiki permite que páginas web alojadas en un servidor público (las páginas wiki) sean escritas de forma colaborativa a través de un navegador, utilizando una notación sencilla para dar formato, crear enlaces, etc, conservando un historial de cambios que permite recuperar de manera sencilla cualquier estado anterior de la página. Cuando alguien edita una página wiki, sus cambios aparecen inmediatamente en la web, sin pasar por ningún tipo de revisión previa.Wiki también se puede referir a una colección de páginas hipertexto, que pueden ser visitadas y editadas por cualquier persona (definición de Wikipedia). Debian tiene su wiki en http://wiki.debian.org/ y FC en http://fedoraproject.org/wiki/ y ambas están basadas en Moin Moin (http://moinmoin.wikiwikiweb.de/). MoinMoin es una Python WikiClone que permite rápidamente inicializar su propia wikiy solo se necesita un servidor de web y el lenguaje Python instalado.

En http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux se encuentran las instrucciones detalladas para instalar Moin Moin sobre De-

bian, pero básicamente se reducen a: 1) Instalar apache2 y mod_python, 2) configurar Apache para apuntar al código de MoinMoin, 3) instalar el paquete moinmoin, 4) configurar moinmoin y 5) reiniciar Apache. Un ejemplo de configuración:

```
apt-get install python-moinmoin
mkdir /var/www/mywiki
cp -r /usr/share/moin/data /usr/share/moin/underlay \
/usr/share/moin/server/moin.cgi /var/www/mywiki
chown -R www-data:www-data /var/www/mywiki
chmod -R g+w /var/www/mywiki
```

- Configurar apache2 añadiendo /etc/apache2/conf.d/wiki (o donde tenga el fichero de configuración):

```
Alias /wiki/ "/usr/share/moin/htdocs/"

<Location /mywiki>
        SetHandler python-program
        PythonPath "['/var/www/mywiki','/etc/moin/']+sys.path"
        PythonHandler MoinMoin.request::RequestModPy.run
        PythonDebug On
</Location>
```

- Reiniciar apache2:

```
/etc/init.d/apache2 reload
```

- Configurar Moinmoin: Editar /etc/moin/farmconfig.py (múltiples wikis)

```
wikis = [
("mywiki", r"^yoursite.com/mywiki/.*$"),
   ]
```

- también se puede usar (solo una wiki):

```
wikis = [
("mywiki", r".*"),
   ]
```

- También en /etc/moin/farmconfig.py quitar el comentario data_dir y data_underlay_dir (uno por cada wiki) y copiar el fichero.

```
cp /etc/moin/moinmaster.py /etc/moin/mywiki.py
```

- Entonces editar /etc/moin/mywiki.py y cambiar:

```
sitename = u'MyWiki'
data_dir = '/var/www/mywiki/data'
data_underlay_dir = '/var/www/mywiki/underlay'
```

La Wiki estará instalada sobre http://yoursite.com/mywiki/

Actividades

1) Configurar un servidor DNS como caché y con un dominio propio.

2) Configurar un servidor/cliente NIS con dos máquinas exportando los directorios de usuario del servidor por NFS.

3) Configurar un servidor SSH para acceder desde otra máquina sin passwd.

4) Configurar un servidor Apache+ SSL+ PHP+ MySQL+ PHPAdmin para visualizar las hojas personales de los usuarios.

5) Crear y configurar un sistema de correo electrónico a través de Exim, fetchmail, Spam-Assassin y un servidor IMAP para recibir correos desde el exterior y poder leerlos desde una máquina remota con el cliente Mozilla (thunderbird).

6) Instalar la Wiki MoinMoin y crear un conjunto de páginas para verificar su funcionamiento.

Otras fuentes de referencia e información

[Debc, LPD03b, Ibi]

http://tldp.org/HOWTO/DNS-HOWTO-7.html

http://tldp.org/HOWTO/NIS-HOWTO/verification.html

Squid proxy server

Proxy Cache: http://www.squid-cache.org/

Transparent Proxy: http://tldp.org/HOWTO/TransparentProxy-1.html

Proftpd: http://www.debian-administration.org/articles/228

PureFtpd: http://www.debian-administration.org/articles/383

Exim: http://www.exim.org/docs.html

Mutt: http://www.mutt.org

ProcMail: http://www.debian-administration.org/articles/242

LWP:http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php

Moin Moin: (http://moinmoin.wikiwikiweb.de/)

Moin Moin + Debian:

http://moinmoin.wikiwikiweb.de/MoinMoinPackages/DebianLinux

Apache2 + SSL: http://www.debian-administration.org/articles/349